Step1 **Remapping component before Loading**

A	export 1 = ?
A+1	export 2 = ?
A+2	export 3 = ?

Relocations

- Set A to be contents of export 3 in new.dll
- Set A+1 to be contents of export 2 in new.dll
- Set A+2 to be contents of export 7 in another.dll

Step 2 **Loading of executable, remapping component and new.dll**

Executable	Remapping Component	new.dll
1000 call 1009	2000 export 1 = ?	3000 export 1 = 3019
1009 jump to address in 1010	2001 export 2 = ?	3001 export 2 = 3006
	2002 export 3 = ?	3002 export 3 = 3027
1010 data = ?	set 2000 to be contents of export 3 in new.dll	
set 1010 to be contents of export 1 in original.dll		3027 <instructions to implement foo()

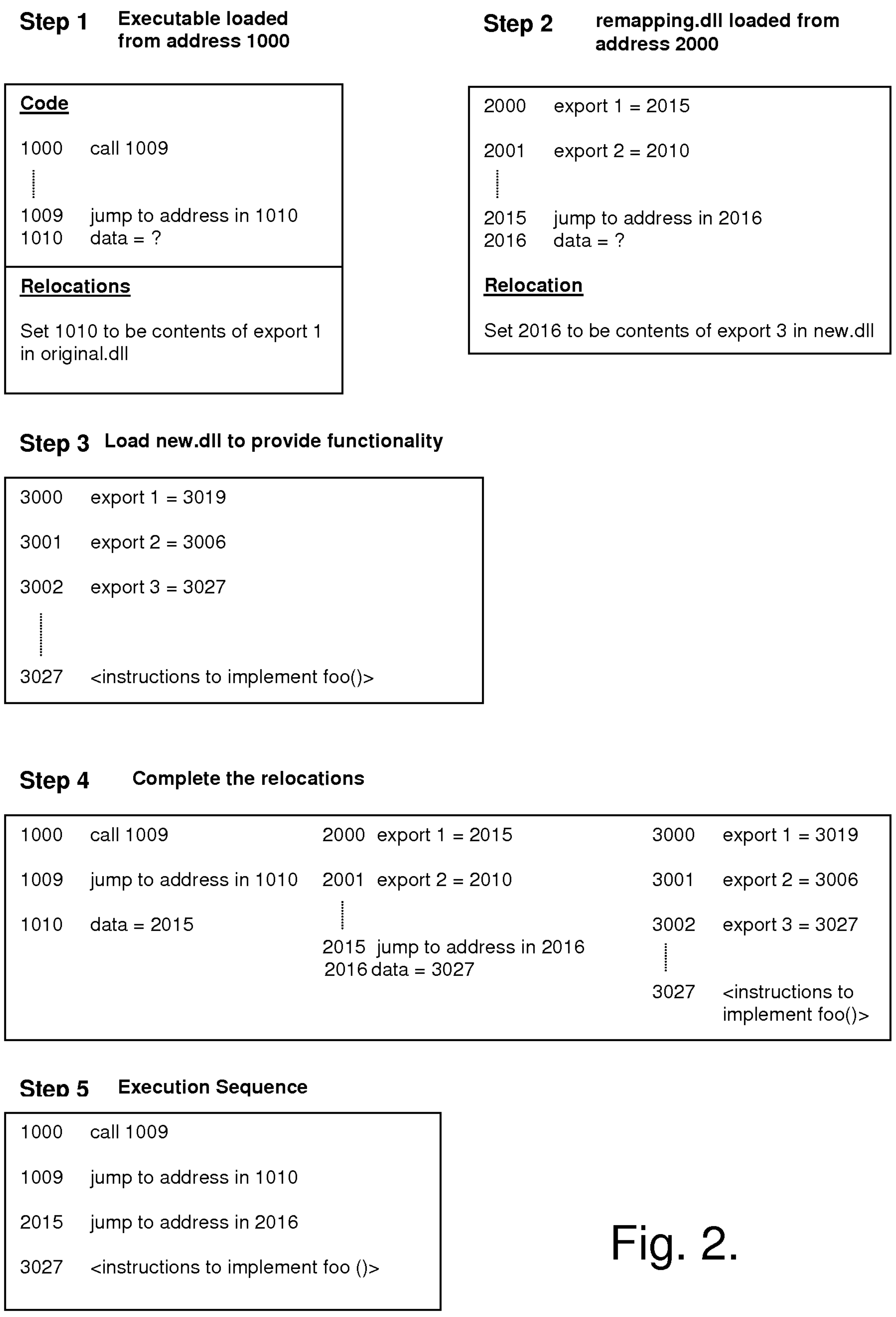
Step 4 **Execution Sequence**

1000	call 1009
1009	jump to address in 1010
3027	<instructions to implement foo()>

Step 3 **Complete the relocations**

Executable	Remapping Component	new.dll
1000 call 1009	2000 export 1 = 3027	3000 export 1 = 3019
1009 jump to address in 1010	2001 export 2 = 3006	3001 export 2 = 3006
	2002 export 3 = 4011	3002 export 3 = 3027
1010 data = 3027		3027 <instructions to implement foo()

Fig. 3.



Step 1 **Executable loaded from address 1000**

```
Code

1000    call 1009
  ⋮
1009    jump to address in 1010
1010    data = ?

Relocations

Set 1010 to be contents of export 1
in original.dll
```

Step 2 **remapping.dll loaded from address 2000**

```
2000    export 1 = 2015

2001    export 2 = 2010
  ⋮
2015    jump to address in 2016
2016    data = ?

Relocation

Set 2016 to be contents of export 3 in new.dll
```

Step 3 **Load new.dll to provide functionality**

```
3000    export 1 = 3019

3001    export 2 = 3006

3002    export 3 = 3027
  ⋮
3027    <instructions to implement foo()>
```

Step 4 **Complete the relocations**

```
1000    call 1009                  2000  export 1 = 2015             3000    export 1 = 3019

1009    jump to address in 1010    2001  export 2 = 2010             3001    export 2 = 3006
                                     ⋮
1010    data = 2015                2015  jump to address in 2016     3002    export 3 = 3027
                                   2016  data = 3027                   ⋮
                                                                     3027    <instructions to
                                                                             implement foo()>
```

Step 5 **Execution Sequence**

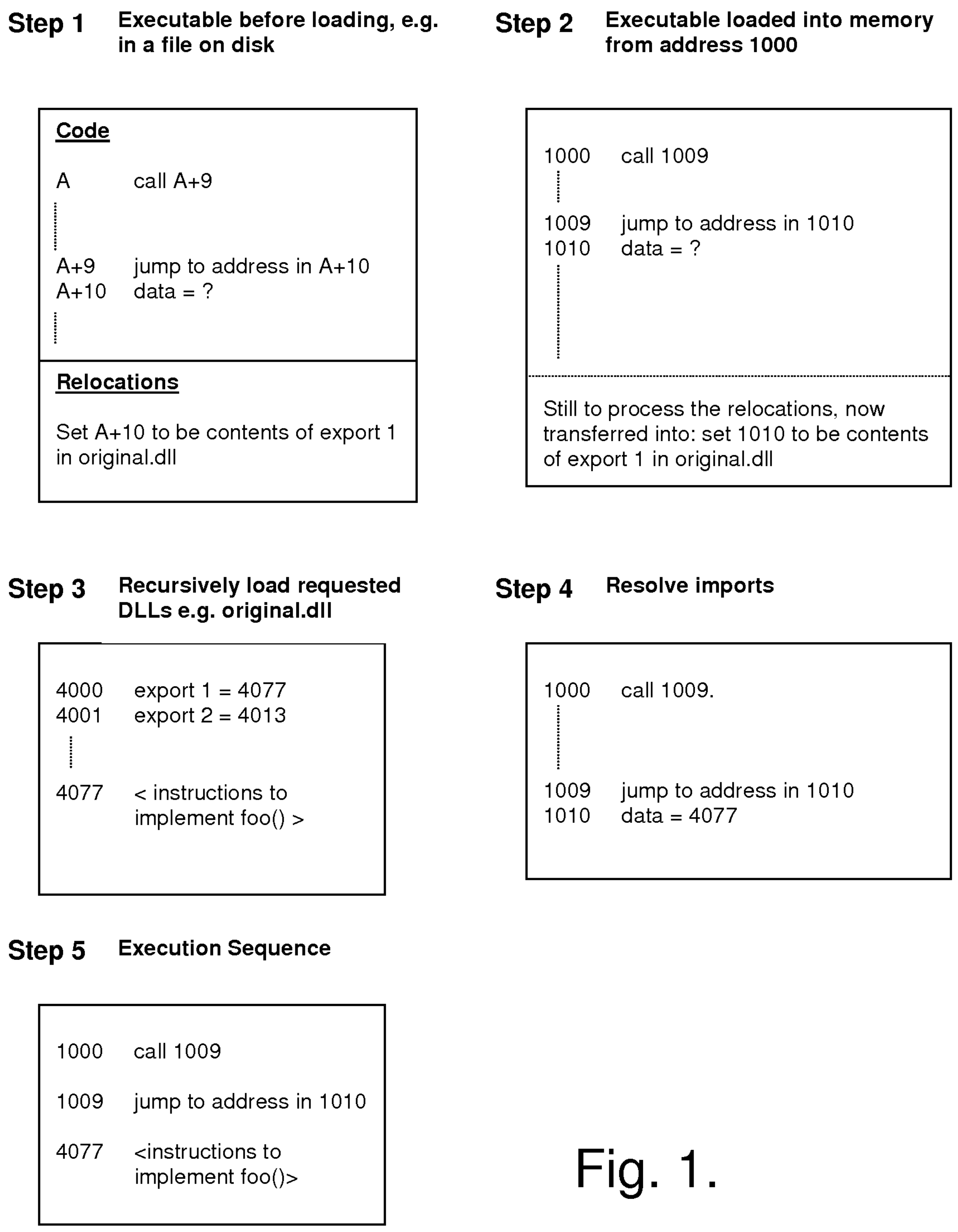
```
1000    call 1009

1009    jump to address in 1010

2015    jump to address in 2016

3027    <instructions to implement foo ()>
```

Fig. 2.



Step 1 **Executable before loading, e.g. in a file on disk**

Code

```
A       call A+9
┆
A+9     jump to address in A+10
A+10    data = ?
┆
```

Relocations

Set A+10 to be contents of export 1 in original.dll

Step 2 **Executable loaded into memory from address 1000**

```
1000    call 1009
┆
1009    jump to address in 1010
1010    data = ?
┆
```

Still to process the relocations, now transferred into: set 1010 to be contents of export 1 in original.dll

Step 3 **Recursively load requested DLLs e.g. original.dll**

```
4000    export 1 = 4077
4001    export 2 = 4013
  ┆
4077    < instructions to
        implement foo() >
```

Step 4 **Resolve imports**

```
1000    call 1009.
┆
1009    jump to address in 1010
1010    data = 4077
```

Step 5 **Execution Sequence**

```
1000    call 1009

1009    jump to address in 1010

4077    <instructions to
        implement foo()>
```

Fig. 1.